**Amendments to the Specification**

Please amend the following paragraphs of the specification as follows:

**[0004]** As technological innovations bring new features (like serial numbers burned into central processing units (CPUs), printers, and monitors) into existing devices as well as completely new types of devices (such as internet protocol (IP)-based telephones), network discovery needs may change. Instead of simply identifying a device on the network, there may be an interest in collecting some additional, presently-unforeseen information about the discovered device. This ever-changing environment requires flexible, easily-reconfigurable and easily-adaptable discovery solutions.

**[0005]** Typically, network discovery is implemented as various forms of exhaustive network sweeps. There are some other approaches based upon querying some auxiliary databases (Doman Name System (DNS), Windows Internet Naming Service (WINS), Dynamic Host Configuration Protocol (DHCP), etc.) to get network addresses of registered devices. However, all these approaches rely on the devices registering themselves in one of those auxiliary databases, which may not occur for all of the devices. Thus, the most authoritative and reliable way for implementing network discovery remains the complete network sweep.

**[0007]** This problem is further exacerbated by the nature of the Transmission Control Protocol/Internet Protocol (TCP-IP) protocol ubiquitous on today's networks. According to the transport level specification of the TCP-IP protocol, a communication between two points on the network is considered failed if an expected response was not received within a certain timeout period. Due to the random nature of signal propagation on TCP-IP networks, the timeout value is typically much larger then the average round-trip time for a message. Thus, to verify an absence

of a device on a particular address in the address space of the network, a discovery agent should send a request and then wait until the expiration of the timeout interval. To improve the reliability of this process in case of a timeout, a discovery agent typically repeats the request 2-3 times, thereby further slowing the discovery process.

**[0008]** Similarly, when a device is discovered on the network, a discovery agent must try various protocols (like Simple Network Management Protocol (SNMP) with different community names, Hypertext Transfer Protocol (HTTP) on different ports, etc.) as the agent does not know ahead of time the nature of the discovered device and on what protocol with which parameters the device will reply. Again, to confirm the failure of a particular protocol, a discovery agent has to wait throughout the timeout interval and then repeat its attempt several times.

**[0009]** Due to the aforementioned problems, typical network discovery solutions are very slow. Existing network management software products (such as HP OPENVIEW ~~like HP OpenView, Visio~~, etc.) would take several days or even weeks to perform an exhaustive sweep of the network. Due to the dynamic nature of networks, discovery data achieved through these processes will be outdated by the time it is collected.

**[0028]** Fig. 1 illustrates an exemplary network on which the present invention can be implemented. The network may comprise a single site 102, or multiple sites 102, 104, 106. Each site 102, 104, 106 may have multiple network resources such as servers 112, 114, 116, and other devices 121, 122, 123, 124, 125, 126, 127, 128, 129 to be discovered. Such devices may be personal computers (PCs), printers, servers, and/or IP-telephones, by way of example. Databases 130, 140, 150 may also be provided for storing data generated in accordance with the

present invention. Alternatively, such data may be stored in servers 112, 114, 116 or on PCs, e.g., device 121. Devices at a single site 102 may be connected over a Local Area Network (LAN). In the multi-site embodiment, the devices may be connected via a Wide Area Network (WAN) or the Internet 108. While Fig. 1 illustrates only a minimal number of network devices, the present invention is usable on a network with any number of devices.

**[0029]** Distributed network resources are tracked and inventoried in accordance with the present invention. In particular, the system and process of the present invention discovers, identifies, and classifies distributed active network resources, as well as collects additional information about the resources using standard IP protocols, such as Internet Control Message Protocol (ICMP), SNMP, HTTP, under the control of custom scripts. A heavily multithreaded agent reaches out to sweep the network discovering active devices using ICMP. Discovered devices are queried for additional information, as specified in the custom scripts, using a combination of SNMP and HTTP protocols. The scripts are exposed to a powerful yet flexible discovery and logging framework while insulating them of the intricacies of the highly efficient discovery process. The information returned through the discovery process is sufficient for proper identification and registration of the discovered devices in a database. Results of the discovery are presented for further processing in the form of industry-standard Extensible Markup Language (XML) files, in the preferred embodiment, although other formats can be used. Agent operations are controlled through a user interface specifically designed to address the needs of network discovery under a variety of requirements. The user interface presents all aspects of managing network discovery using the present invention in a highly-structured, comprehensive

way, which allows for direct navigation to the appropriate aspects of the management infrastructure.

**[0044]** In the preferred embodiment, the manager object 300 is a Component Object Model (COM) object that provides an easy Automation interface into a pool of worker threads that carry out individual discovery operations. As the manager object 300 fully complies with the Automation interface, it can be used with any Automation container (such as VISUAL BASIC, EXCEL VBA ~~Visual Basic, Excel VBA~~) to provide fully asynchronous access to the parallel operations carried out by the threads in the worker thread pool. This allows the discovery agent, in the preferred embodiment, to be implemented in MICROSOFT's VISUAL BASIC ~~MS Visual Basic~~ without the typical loss of performance as compared to Visual C++.

**[0052]** Signal thread 408 ~~402~~ pops processed requests from the result queue 406, analyzes their status and requested operation, and posts them to the original calling thread, which invoked the corresponding method on the manager object 300, through the appropriate event procedure. To protect the calling thread from overruns, the signal thread 408 ~~402~~ switches context to the original calling thread context prior to posting the event. The switching context synchronizes the signal thread 408 ~~402~~ with the calling thread for the period of posting the event back to the calling thread, thus providing for the COM Automation-compatible mechanism for posting back asynchronous events.